mo_acc_util subroutine module

User Guide

Sergey Sukov[1]

[1] Forschungszentrum Jülich GmbH, Jülich, Germany

Contact: s.sukov@fz-juelich.de; info@nat-esm.de

Published on 07.01.2025 on https://www.nat-esm.de/services/accepted-sprints

Module **mo_acc_utils** contains a set of subroutines for comparing the real (single or double precision) values of 1D-5D matrix elements on CPU and GPU using OpenACC features. Data comparison subroutines have uniform prototypes with the same number of arguments:

```
SUBROUTINE accCompareRealMatrixHostDevice1D(matrix, matrixName, &
                                            & deviation, detailedOutput)
REAL(_REAL_TYPE_), DIMENSION(:), POINTER, INTENT(IN) :: matrix
CHARACTER(len=*), INTENT(IN) :: matrixName
REAL(_REAL_TYPE_), OPTIONAL, INTENT(OUT) :: deviation
LOGICAL, OPTIONAL, INTENT(IN) :: detailedOutput
.............................................................
.............................................................
.............................................................
END SUBROUTINE accCompareRealMatrixHostDevice1D
```

The input parameters passed to the subroutine are the pointer **matrix** associated with the matrix memory on the CPU, the string **matrixName** containing its name, and the optional logical flag **detailedOutput**, which controls the output of detailed information messages to the screen. The maximum absolute difference between the values of the matrix elements on the CPU and GPU is stored to the **deviation** variable (if optional argument present). The specific size of real data type is set in the header section of the source code file **mo_acc_util.f90** using a preprocessor directive

```
#define _REAL_TYPE_ 8
```

Special ICON settings are activated by the directive

```
#define _ICON_STYLE_
```

To automatically call a subroutine corresponding to the matrix shape, the user is provided with the overloaded interface **accCompareRealMatrixHostDevice**:

```
USE mo_acc_util, ONLY: accCompareRealMatrixHostDevice
.....................................................
REAL(8), ALLOCATABLE, TARGET :: matrix3D(:,:,:), deviation
REAL(8), DIMENSION(:,:,:), POINTER :: ptrMatrix3D
.....................................................
ptrMatrix3D => matrix3D
CALL accCompareRealMatrixHostDevice(ptrMatrix3D, 'Matrix 3D', &
                  & detailedOutput=.TRUE., deviation=deviation)
.....................................................
```

During the subroutine execution, either brief (**detailedOutput = .FALSE.**, default) or detailed (**detailedOutput = .TRUE.**) information messages are displayed on the screen (**stderr** stream). The detailed output includes the name of the called subroutine, the matrix name, a description of the matrix shape with the lower and upper bounds along each dimension, the ranges

of matrix element values on the CPU and GPU, and the maximum absolute difference between the values:

```
>>>>> accCompareRealArrayHostDevice3D
>>>>> MATRIX NAME: Matrix 3D
>>>>> MATRIX SHAPE: (7:239) (44:98) (-7:12)
>>>>> MIN/MAX: CPU [0.000000E+00;0.256299E+04]
      GPU [0.000000E+00; 0.256299E+04] DEVIATION 0.000000E+00
```

The brief output consists of only one string with the matrix name, the ranges of its element values, and the deviation:

```
>>>>> Matrix 3D: CPU [0.000000E+00;0.256299E+04]
      GPU [0.000000E+00;0.256299E+04] DEVIATION 0.000000E+00
```

The absence (or partial presence) of a matrix copy on the GPU is considered a fatal error. In this case, program execution is interrupted using the **STOP** statement. An error message is displayed on the screen with the number of memory cells allocated to the GPU:

```
>>>>> accCompareRealArrayHostDevice3D
>>>>> GPU DATA PARTIALLY PRESENT
>>>>> MATRIX NAME: Matrix 3D
>>>>> MATRIX SHAPE: (7:239) (44:98) (-7:12)
>>>>> MATRIX SIZE: 256300
>>>>> GPU PRESENT: 194837
```

In this example, the 3D matrix consists of 256300 elements, but memory for only 194837 of its elements is allocated on the GPU.

When calling the module's subroutines from the CPU programs (OpenACC is disabled), information about the shape of the matrices and the intervals of their element values is displayed on the screen:

```
>>>>> accCompareRealArrayHostDevice3D
>>>>> MATRIX NAME: Matrix 3D
>>>>> MATRIX SHAPE: (7:239) (44:98) (-7:12)
>>>>> MIN/MAX: CPU [0.000000E+00;0.256299E+04] GPU NOT USED
```

or

```
>>>>> Matrix 3D: CPU [0.000000E+00;0.256299E+04] GPU NOT USED
```

The source code archive contains four files:

- **mo_acc_util.f90** - main module source code file;
- **main.f90** - emulator program code with examples of calling module subroutines;
- two makefiles (**Makefile**, **MakefileCPU**) for building emulator program CPU and GPU executables using nvhpc compiler. Before compiling the code for GPU (**Makefile**), you need to set (or just leave empty corresponding field) the correct variables with the CUDA version (e.g., **CUDAVER = -gpu=cuda12.3**) and device architecture (e.g., **SMARCH = -gpu=sm_86**).